Engineer

カナリアリリース

新版を実際のトラフィックの一部で試し、段階的に広げる公開方法

どのようなものか

カナリアリリースは、まず、サービスの実際のトラフィックの小さな割合だけに新版を提供し、置き換えられる旧版で、それ以外のすべてを処理し続ける方法です。新版が問題なく動けば、段階的に割合を増やし、最後にすべてを任せ、旧版を廃止します。うまく動かなければ、停止していなかった旧版へトラフィックを戻し、不具合を経験するのは、その小さな割合だけで済みます。Danilo Satoは、2014年6月にMartin Fowlerのblikiでこの手法を説明し、phased rollout（段階的な展開）やincremental rollout（少しずつ進める展開）とも呼ばれると記しています。Kubernetesのツールでは、カナリアデプロイ（canary deployment）と呼ぶことが多く、この種のリリース戦略を追加するKubernetesコントローラー、Argo Rolloutsは、設定名を `canary` としています。

名前は炭鉱に由来します。鉱夫は、籠に入れたカナリアを地下へ連れて行きました。漏れたガスの影響が人に及ぶ前に鳥が弱るため、その異変が、外へ出る警告になったからです。トラフィックの小さな割合も、同じ役割を果たします。まだ対処する時間がある間に、先に問題に遭遇します。

その割合に誰を含めるかは、選択です。Satoは、無作為に抽出した利用者、一般公開前の自社の従業員、プロファイルや人口統計的な属性で選ぶ利用者を挙げています。大きなシステムでは、代わりに、すでにある区分を使い、一つの地域、拠点、ブランドから始めることもよくあります。Satoの説明によると、Facebookは複数のカナリアを使い、最初のものは従業員だけに公開し、すべての機能フラグを有効にして、新機能の問題を早く見つけていました。

ブルーグリーンデプロイとの違いは、移し方です。ブルーグリーンは、二つ目の完全なコピーを構築し、一度の切り替えで全トラフィックを移し、旧コピーを戻る先として残します。カナリアは、トラフィックを徐々に移すため、しばらくの間、二つのバージョンが並んで実際の利用者へ応答します。ローリングアップデ

ートも、二つのバージョンを同時に動かしますが、準備が整う速さに合わせてインスタンスを置き換え、新版の調子を判断するために停止しません。カナリアは、各段階で止まり、その判断を待ちます。

概念を判断に生かす

「カナリアリリースとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 変更を検証し、リリースし、元に戻せる仕組みを整え、金曜の深夜ではなく火曜の午後にお客様へ届けられるようにします。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 GitHub ActionsでGIS SaaSのデプロイを自動化 — デプロイ、データ、報告を手作業から解放

GitHub ActionsでGIS SaaSのデプロイを自動化 お問い合わせ →

なぜ重要なのか

本番で、実際の負荷、データ、クライアントの組み合わせにさらされて初めて現れる不具合があります。ステージング環境は近づけられても、同一にはなりません。カナリアは、それを受け入れ、代わりに被害を限定します。二十件に一件が新版へ届くなら、悪いリリースの影響は、気付くまでの間、二十件に一件に限られ、対処は、その分を旧版へ戻すことです。

Satoは、二つ目の用途として、処理能力のテストを挙げます。割合をゆっくり増やせば、安全に戻す手段を持ちながら、本番の負荷が増えたときの新版の動作を確認できます。負荷テスト専用の環境を別に構築しても、本番そのものほど本番に近くはできません。

ただし、カナリアの良し悪しは、それを判断する信号に依存します。エラー、遅延、登録・支払い・注文などの重要な業務数値が、バージョンごとに、次の段階を止められるだけ早く見える必要があります。それがなければ、カナリアは、待ち時間を増やした遅い展開にすぎません。Satoは、IMVUが、業務指標を監視し、統計的に有意な悪化があれば自動でリリースを戻す、「クラスタの免疫システム」と呼ばれる仕組みを先駆けて実践したと記しています。

運用上の負担もあります。二つ以上のバージョンが同時に稼働し、その間、共有するデータベース、キャッシュ、キュー、クライアントが呼ぶAPIのすべてが、両バージョンに対応する必要があります。Satoは、同時に動かすバージョン数を最小限にするよう助言します。また、利用者自身のパソコンやスマートフォンへインストールするソフトウェアには、あまり適しません。提供者が、更新の時期

を決められないからです。その場合は、バックエンドを両方に対応させ、使われているクライアントのバージョンを監視し、旧版の割合が十分に下がってから廃止することを勧めています。

実際の活用例

Kubernetesでの最も単純なカナリアは、安定版のPodを選ぶServiceと、カナリアのPodを選ぶServiceの二つと、その間でトラフィックを重み付けして送る経路です。Gateway APIでは、重みをHTTPRouteに記述します。

```
apiVersion: gateway.networking.k8s.io/v1
kind: HTTPRoute
metadata:
  name: web
spec:
  parentRefs:
    - name: prod-web
  hostnames:
    - "www.example.com"
  rules:
    - backendRefs:
        - name: web-stable
          port: 8080
          weight: 95
        - name: web-canary
          port: 8080
          weight: 5
```

重みは、パーセントではなく比率です。各バックエンドに届く割合は、その重みをルール内の全重みの合計で割った値であるため、95と5なら、二十件に一件をカナリアへ送ります。重みの省略は1として扱い、0なら何も送りません。
Gateway APIのトラフィック分割ガイドは、要求ヘッダーを照合するルールも示しています。例では `traffic: test` で、その要求だけを新版へ送り、顧客より先にテスト担当者がアクセスできます。カナリアを先へ進めるには二つの数値を編集し、引き戻すには、その重みを0にします。

Argo Rolloutsは、段階的な進行を自動化します。 `Deployment` を、同じPodテンプレートを持つ `Rollout` で置き換え、テンプレートが変わるたびに、コントローラーがたどる手順を列挙します。

```
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: web
spec:
  replicas: 10
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
        - name: web
          image: registry.example.com/web:1.4.2
  strategy:
    canary:
      maxSurge: "25%"
      maxUnavailable: 0
      steps:
        - setWeight: 10
        - pause:
            duration: 1h
        - setWeight: 20
        - pause: {}
```

トラフィックルーターがなければ、ArgoはPod数で重みを近似します。レプリカが十個で10パーセントなら、カナリアが一つ、安定版が九つです。最初の一時停止は一時間待ち、二つ目は、誰かが判断するまで待ちます。リリースと、その二つの結末は、次のようになります。

```
kubectl argo rollouts set image web web=registry.example.com/web:1.5.0
# 期限のない一時停止で、続行する場合は…
kubectl argo rollouts promote web
# …または中止して、全トラフィックを安定版へ戻す
kubectl argo rollouts abort web
```

この一連の処理すべては、一時停止の間に、カナリアが旧版より悪く動いていないか分かることに依存します。それが、不具合がまだ小さいうちに通知するアラートを中心とした、サイト信頼性・監視サービスが担う仕事です。2025年からの職務で手がけた自社管理のエラー監視とトレースは、その信号の例を示します。エラーを一件ずつ並べる代わりに、繰り返し発生する問題ごとにまとめ、急増や悪化を検出し、通知にクールダウン期間を設け、一度の悪いデプロイで四

十回も緊急通知しないようにします。カナリアの各段階に必要なのは、まさにそれです。翌朝、利用者が偶然見つける悪化ではなく、一時停止の間に分かる悪化です。

問題が起きるところ

カナリアが小さすぎて、何も判断できません。 閑散時の十個のPodのうち一つでは、一時停止中に数百件しか処理しないことがあり、千件に一件の不具合が現れないかもしれません。測るべきなのは経過時間ではなく、カナリアが処理した要求数です。発生頻度の低いエラーも複数回現れ得るよう、割合と一時停止時間を一緒に決め、夜間は正午より長く、その段階を維持します。

ダッシュボードが、カナリアと安定版を区別できません。 カナリアが、安定版より五十件に一件多く失敗していても、割合が5パーセントなら、全体の失敗は、約千件に一件増えるだけです。サービス全体の一つのエラー率しか示さないダッシュボードでは、見えません。カナリアと安定版のPodに異なるラベルを付け、指標にも引き継ぎ、固定のしきい値ではなく、安定版と比較します。Argo Rolloutsでは、`canaryMetadata` と `stableMetadata` でラベルを追加できます。Rolloutのバックグラウンド `analysis` は、失敗すると更新を中止します。

両バージョンが、同じデータへ書き込みます。 展開の間、新旧のコードは同じデータベースを読み書きし、無作為に振り分けると、一人の利用者の要求も、一分以内に両方へ届くことがあります。旧版が値を入れない新しい列や、読み取れない形式が、安全であるはずの安定版のエラーになります。Satoの答えは、並行した変更です。展開の間ずっと、データベースが両方のアプリケーションに対応し、旧版がなくなってから、古い形を削除します。

新版へのトラフィックは止まっても、変更は残ります。 `kubectl argo rollouts abort` は、トラフィックを安定版のReplicaSetへ戻しますが、Rolloutのspecは、新しいイメージを指定したままです。再開すると、そのイメージが再び対象になります。ロールバックが完了するのは、テンプレートを旧版へ戻したときだけです。specをリポジトリで保持しているなら、そこを変更し、何を動かすべきかについて、ソースとクラスタを一致させます。

カナリアを、A/Bテストにも使います。 経路の振り分け方が同じに見えるため、カナリアのトラフィックで、新機能がコンバージョンへ与える効果も測りたくなります。Satoは、二つを分けるよう助言します。カナリアは問題を探し、数分から数時間で終えるべきものです。一方、実験は仮説を検証し、統計的に有意な結

果に達するまで、何日分ものデータが必要かもしれません。業務指標でカナリアを判定すると、実験で生み出そうとする効果も、判定を曖昧にします。カナリアでリリースしてから、機能フラグなど、独自の分割の仕組みで実験を行います。

参考資料

- Canary Release：Martin Fowlerのblikiに掲載された、Danilo Satoによる2014年の説明です。利用者の選び方、処理能力のテスト、A/Bテスト、データベースを扱います。
- Argo Rollouts：Canary：手順と一時停止、Pod数で近似する重み、トラフィックルーターと組み合わせたスケーリングを説明します。
- HTTPトラフィックの分割：重み付きバックエンドと、ヘッダーに一致するカナリア経路についてのGateway APIのガイドです。
- kubectl argo rollouts abort：中止操作がトラフィックに与える影響と、specに残るものを説明します。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「カナリアリリースとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く 連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/canary-release/